**1.Code Generation & Debugging**

**Set-1**

**Task:**  
AI (ChatGPT/Copilot) generated a **multi-file project**:

* **Vehicle.cpp/h** → Defines a Vehicle class (**id, speed, temperature, fuel**).
* **FleetManager.cpp/h** → Manages vector of vehicles, computes averages, and triggers alerts.
* **main.cpp** → Loads vehicle data from CSV and prints results.

**Program Features:**

* Load vehicle data from CSV.
* Compute average speed, temperature, and fuel.
* Print **alerts**:
  + Temp > 110 °C → "**Critical Overheating**"
  + Fuel < 15% → "**Low Fuel Warning**"

**2. Code Review & Refactoring**

Messy code snippet (single file, no OOP):

**#include <iostream>**

**#include <fstream>**

**#include <vector>**

**using namespace std;**

**int main(){**

**vector<int> speed = {120, 80, 0};**

**vector<int> temp = {130, 90, 85};**

**vector<int> fuel = {10, 40, 50};**

**for(int i=0;i<3;i++){**

**cout<<"Vehicle "<<i<<" Speed:"<<speed[i]<<" Temp:"<<temp[i]<<" Fuel:"<<fuel[i]<<endl;**

**if(temp[i]>110) cout<<"Alert"<<endl;**

**if(fuel[i]<15) cout<<"Warning"<<endl;**

**}**

**}**

**Tasks:**

* **Review Issues:** no OOP, magic numbers, no input validation, no modularization, hardcoded arrays.
* **Refactor into OOP design:**
  + **class Vehicle { … }**
  + **class FleetManager { … }**
* Use **encapsulation**, **constructors**, **const correctness**, and **exception handling**.
* Implement file input with proper error checks.

**3. Unit & Integration Testing**

* Use **GoogleTest or Catch2 or cassert** inside VS Code.
* **Tests to implement:**
  + Vehicle with **temp=120** → “**Critical Overheating**”
  + Vehicle with **fuel=10** → “**Low Fuel Warning**”
  + Fleet **[80, 90, 100]** speeds → avg = 90
  + Boundary: exactly 110 → “**Critical Overheating**”, exactly 15% fuel → No Warning
  + Empty dataset → must throw exception

**4. Documentation**

* **README.md** (write this in VS Code):
  + Introduction ,System Requirements (g++, VS Code extensions)
  + Explanation of classes/functions, Sample input/output

**5. Concurrency**

* Use **std::thread** in C++ to simulate multiple vehicles updating speed/temp/fuel concurrently.
* Use **std::mutex** to prevent race conditions when FleetManager aggregates.

Compare single-thread vs multi-thread performance.

**List of Prompts are given to Github Copilet**

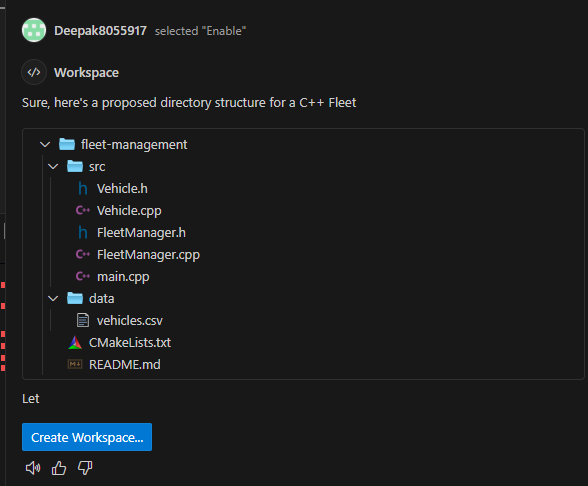
**Step 1) Code genration or Creating a Workspace**

@workspace /new Gnerate the clena work space for my project in cpp in multi file system  
below are list of required ments

Vehicle.cpp/h → Defines a Vehicle class (id, speed, temperature, fuel).  
FleetManager.cpp/h → Manages vector of vehicles, computes averages, and triggers alerts.  
main.cpp → Loads vehicle data from CSV and prints results.

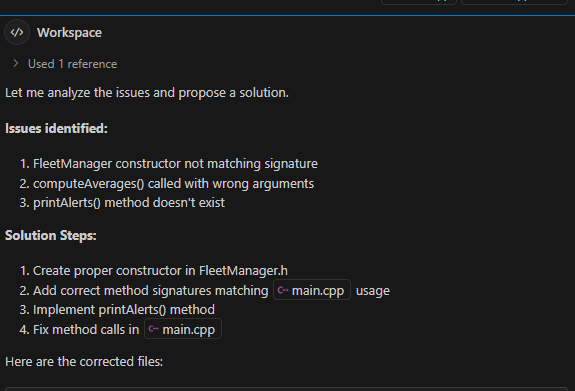
follow the above structre and genrate the codes  
and the code feature are :  
Load vehicle data from CSV.  
Compute average speed, temperature, and fuel.  
Print alerts:  
Temp > 110 °C → "Critical Overheating"  
Fuel < 15% → "Low Fuel Warning"

this is the strtcutre okay genrate now



**Step 2) Code Debugging :**

@workspace /fix #file:main.cpp here i am facing some issue between line no 34 to 36 please check the dependencies and reslove the isuue and check the pitfalls



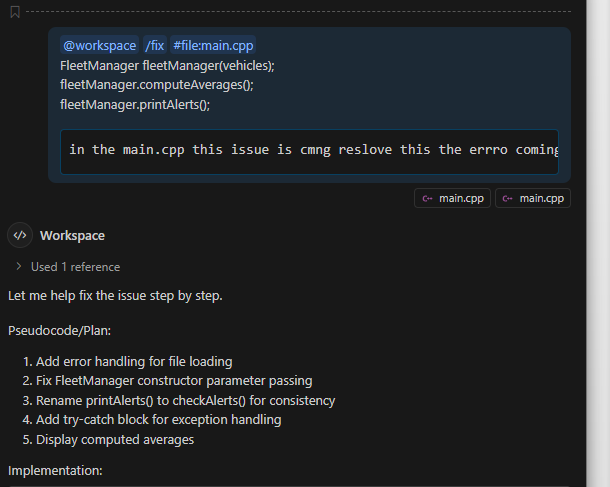
**Step 2.1) Debugging and Fixing the bugs:**

@workspace /fix #file:main.cpp

FleetManager fleetManager(vehicles);

fleetManager.computeAverages();

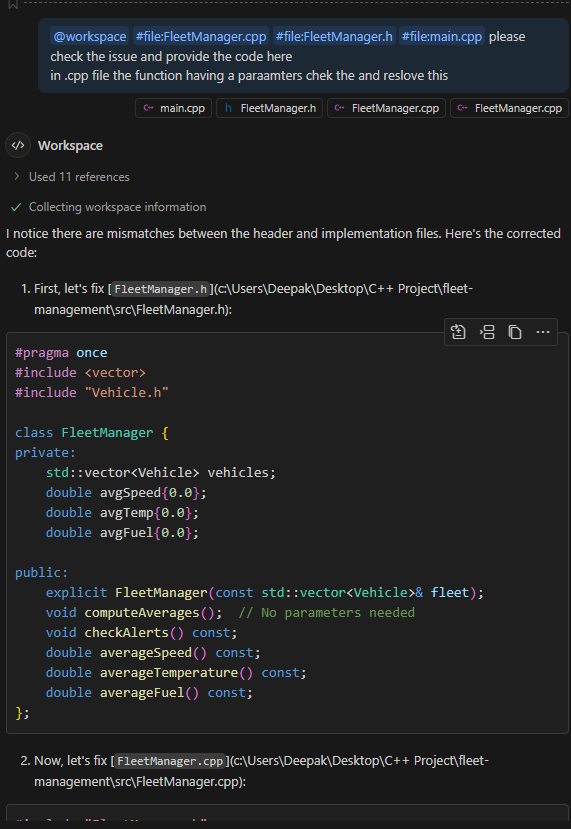
fleetManager.printAlerts();

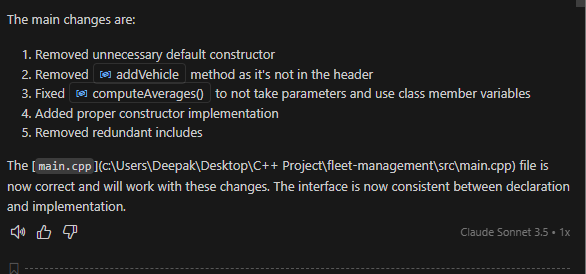


**Step 2.2 ) Facing Issues at Constrcter:**

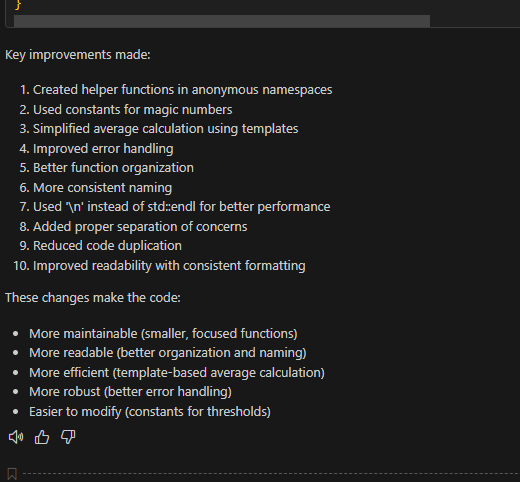
@workspace #file:FleetManager.cpp #file:FleetManager.h #file:main.cpp please check the issue and provide the code here

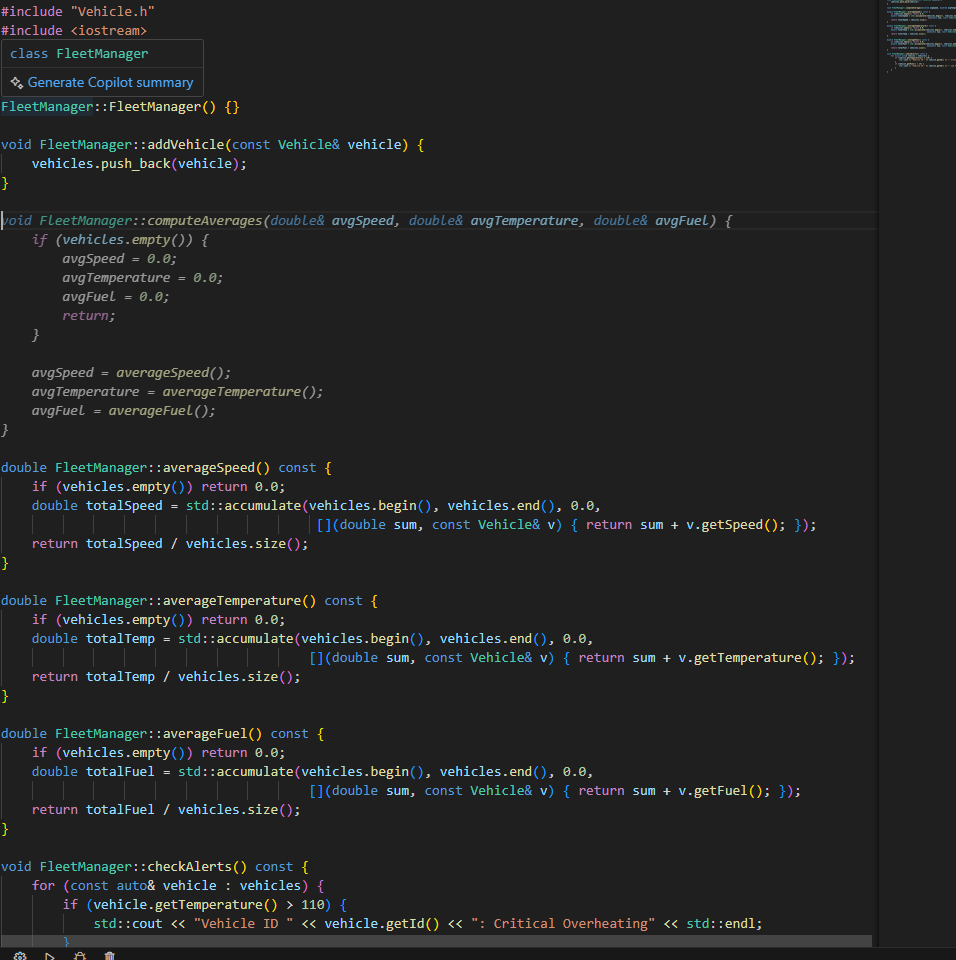
in .cpp file the function having a paraamters chek the and reslove this





**Step 3: Code Refractoring**  
  
@workspace #file:FleetManager.cpp Here this file has more no of lines without changing any functionlaity of the code i wnat refractor the above code okay

#file:main.cpp and this code also it has good for both redability and maintinability  
  
  




**Step 4) Ouput or Execution (commnds) :**

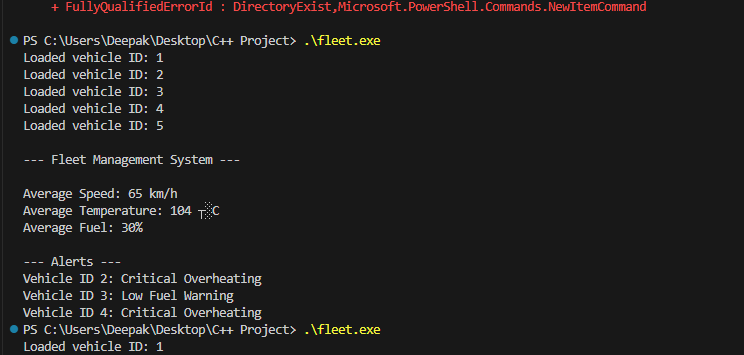
@workspace PS C:\Users\Deepak\Desktop\C++ Project> .\fleet.exe

Error: Unable to open file: data/vehicles.csv

PS C:\Users\Deepak\Desktop\C++ Project>

no tgetting ouput please this the ouput has to print in the terminal and the dta has to fecth from the data/vehicles.csv

please provide



**Step 5) Unit Test cases:**  
  
@workspace /tests Now genrate the unit test for this file here i am using Catch2 okay genrat the unit test cases for that here i will give some exmple type unit test

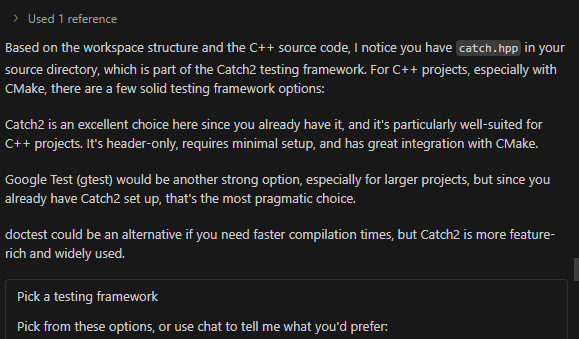
1)Vehicle with temp=120 → “Critical Overheating”

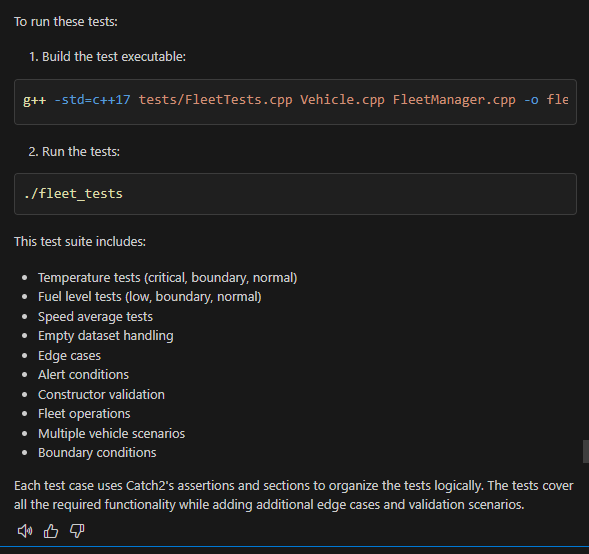
2)Vehicle with fuel=10 → “Low Fuel Warning”

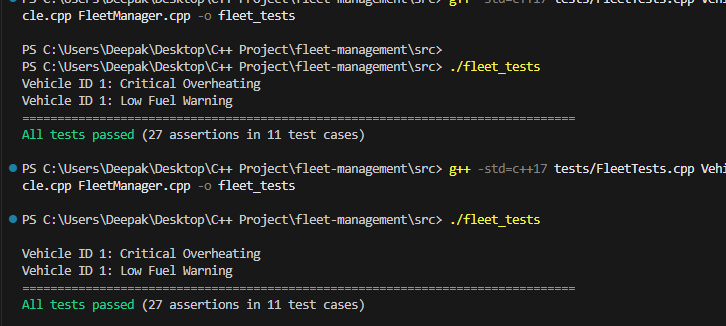
3)Fleet [80, 90, 100] speeds → avg = 90

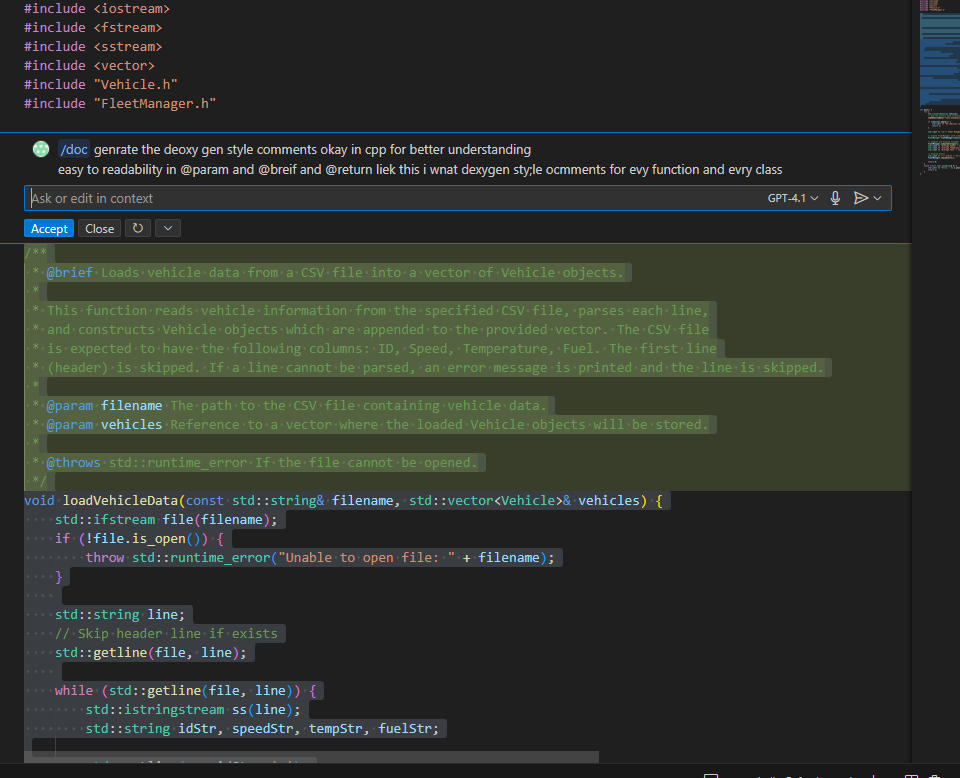
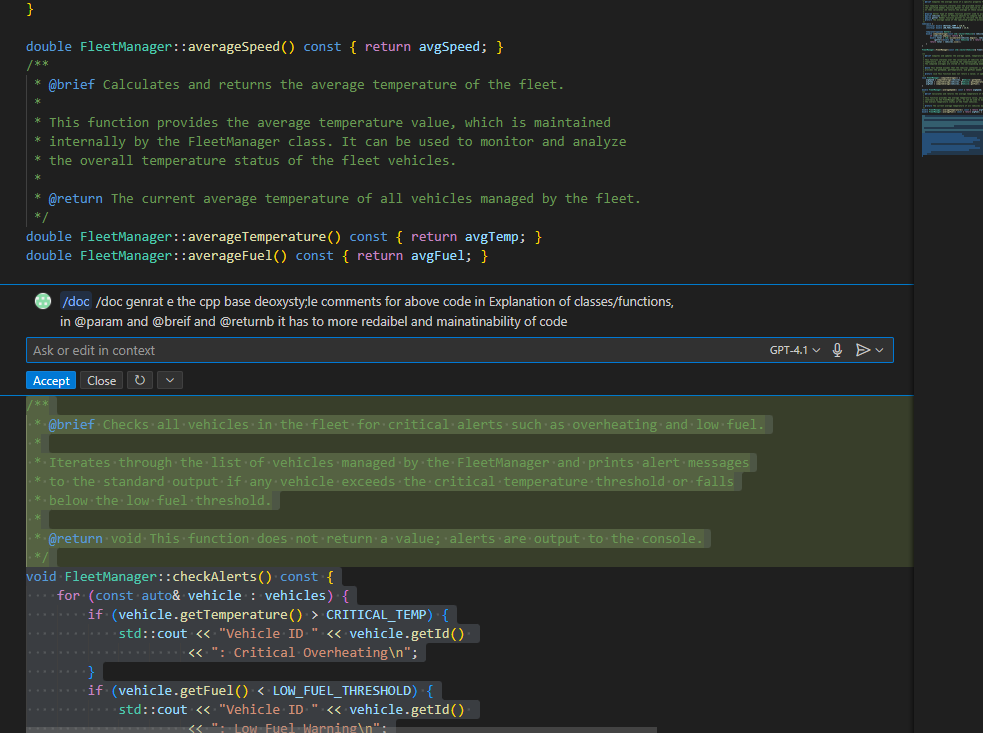
4)Boundary: exactly 110 → “Critical Overheating”, exactly 15% fuel → No Warning

5)Empty dataset → must throw exception

gnerate the test cases file in the work directry in the /src folder okay here i wnat 20 unit test cases genrate now  
  
  




**Output of Unit Test cases**   
  


**Code Documentation**  
  


**Upto Here the give Task is Complted  
now   
Task 2:**  
  
  
#file:task2.cpp in this code there is no proper strcutre and this code os not following any oops and it i procedure oreinted one and check the isuues no OOP, magic numbers, no input validation, no modularization, hardcoded arrays.

now prvide in the oops format and highly secure and proper refracting

Follow this steps

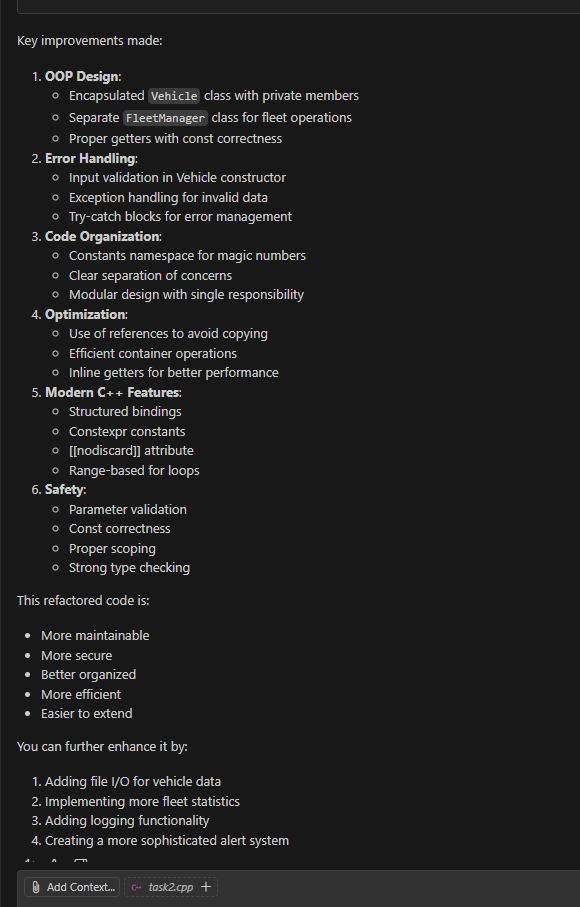
Refactor into OOP design:

class Vehicle { … }

class FleetManager { … }

Use encapsulation, constructors, const correctness, and exception handling.

add execptions for error handling and Implement file input with proper error checks.

make the code most optmized way that run in less time complexity  
  
**Code review**

#file:task2.cpp do the code for the this task2.cpp file and catch the possible bugs and is there nay issues or error are presnt in the code

and make it check is there any pit falls presnet in the code  
  
